JESSICA LIM 31954081

FIT3152 : ASSIGNMENT 2

Friday **6/19/2022** **ASSIGNMENT 2**

```
> summary(WAUS)
      Day              Month             Year         Location      MinTemp          MaxTemp         Rainfall       Evaporation    
 Min.   : 1.00   Min.   : 1.000   Min.   :2007   Min.   : 2.00   Min.   :-6.80   Min.   : 4.90   Min.   : 0.000   Min.   :  0.000  
 1st Qu.: 8.00   1st Qu.: 4.000   1st Qu.:2011   1st Qu.: 5.00   1st Qu.: 6.50   1st Qu.:17.18   1st Qu.: 0.000   1st Qu.:  2.600  
 Median :16.00   Median : 7.000   Median :2013   Median :10.00   Median :10.50   Median :21.60   Median : 0.000   Median :  4.800  
 Mean   :15.85   Mean   : 6.519   Mean   :2014   Mean   :15.16   Mean   :10.62   Mean   :22.64   Mean   : 1.775   Mean   :  5.994  
 3rd Qu.:24.00   3rd Qu.: 9.000   3rd Qu.:2017   3rd Qu.:21.00   3rd Qu.:14.60   3rd Qu.:27.40   3rd Qu.: 0.400   3rd Qu.:  8.000  
 Max.   :31.00   Max.   :12.000   Max.   :2019   Max.   :44.00   Max.   :28.90   Max.   :45.20   Max.   :82.200   Max.   :133.900  
 NA's   :16      NA's   :15       NA's   :19                     NA's   :25      NA's   :20      NA's   :57       NA's   :916      
    Sunshine      WindGustDir        WindGustSpeed   WindDir9am        WindDir3pm        WindSpeed9am    WindSpeed3pm   
 Min.   : 0.000   Length:2000        Min.   :11.0   Length:2000        Length:2000        Min.   : 0.00   Min.   : 0.00  
 1st Qu.: 4.700   Class :character   1st Qu.:31.0   Class :character   Class :character   1st Qu.: 7.00   1st Qu.:13.00  
 Median : 8.500   Mode  :character   Median :39.0   Mode  :character   Mode  :character   Median :13.00   Median :19.00  
 Mean   : 7.668                      Mean   :40.7                                         Mean   :14.82   Mean   :18.69  
 3rd Qu.:10.800                      3rd Qu.:48.0                                         3rd Qu.:20.00   3rd Qu.:24.00  
 Max.   :13.600                      Max.   :96.0                                         Max.   :57.00   Max.   :54.00  
 NA's   :977                         NA's   :261                                          NA's   :40      NA's   :111    
  Humidity9am      Humidity3pm      Pressure9am      Pressure3pm        Cloud9am        Cloud3pm        Temp9am          Temp3pm      
 Min.   :  2.00   Min.   :  1.00   Min.   : 989.2   Min.   : 989.4   Min.   :0.000   Min.   :0.0   Min.   :-0.30   Min.   : 3.20  
 1st Qu.: 55.00   1st Qu.: 30.00   1st Qu.:1013.6   1st Qu.:1011.3   1st Qu.:1.000   1st Qu.:2.0   1st Qu.:11.50   1st Qu.:15.90  
 Median : 70.00   Median : 47.00   Median :1018.5   Median :1016.0   Median :6.000   Median :5.0   Median :15.10   Median :20.10  
 Mean   : 67.17   Mean   : 47.19   Mean   :1018.4   Mean   :1016.0   Mean   :4.458   Mean   :4.6   Mean   :15.78   Mean   :21.23  
 3rd Qu.: 83.00   3rd Qu.: 63.00   3rd Qu.:1023.4   3rd Qu.:1020.7   3rd Qu.:7.000   3rd Qu.:7.0   3rd Qu.:19.50   3rd Qu.:26.00  
 Max.   :100.00   Max.   :100.00   Max.   :1038.9   Max.   :1035.9   Max.   :8.000   Max.   :8.0   Max.   :36.80   Max.   :43.40  
 NA's   :40       NA's   :94       NA's   :45       NA's   :40       NA's   :711     NA's   :810   NA's   :25      NA's   :98     
 WarmerTomorrow        Date          
 Min.   :0.0000   Min.   :2007-11-21  
 1st Qu.:0.0000   1st Qu.:2011-03-18  
 Median :1.0000   Median :2013-11-20  
 Mean   :0.5521   Mean   :2014-02-20  
 3rd Qu.:1.0000   3rd Qu.:2017-07-01  
 Max.   :1.0000   Max.   :2019-12-27  
 NA's   :31       NA's   :49          
```

1. **Explore the data: What is the proportion of days when it is warmer than the previous day compared to those where it is cooler? Obtain descriptions of the predictor (independent) variables – mean, standard deviations, etc. for real-valued attributes. Is there anything noteworthy in the data? Are there any attributes you need to consider omitting from your analysis?**

Above is the result of the Summary we got from the WAUS. We can see the result of the minimal, first quartal, median, mean, third quartal, maximum and the number of NA's each inside of all the variables.

THE PROPORTION OF THE WARMER

We can see from this calculated below that the proportion of the warmer days is around 54.35%.

```
today_warmer_percentage <- (count(today_warmer) / count(WAUS)*100)
today_warmer_percentage
    n
54.35
```

I got this data by counting the "WarmerTomorrow == 1" from the WAUS data. And from that, we got the total of those who have 1's divided by the total data of the WAUS. I got this because I thought that If tomorrow is warmer then tomorrow is equal to the day where the previous day is cooler.

From the picture above and on the side we can see the basic information like min, max, median, quartal, sd, var, range, and many others.

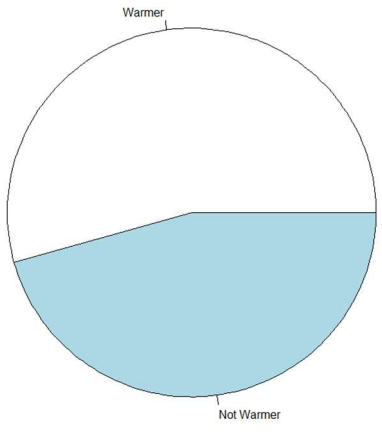
After observing the data, we can also see that there is some noteworthy information from the data. This leads us to the next question which asks us about any noteworthy things, in this case, we can see in the summary that there are a lot of NAs in the data.

When there are a lot of NAs in the data we can consider there will be a lot of problems arising. It can influence the result of the data.

Data loss can occur for many reasons, and it is worth considering whether the loss introduces a bias in the predictive model. Therefore, we need to clean up the data so that the data is better and we can achieve a better result. To do this we can just drop all the column who has NAs. I use this:

```
#drop the NA's
WAUS <- WAUS[rowSums(is.na(WAUS)) == 0,]
```

So now that we have dropped the NAs. The next question is if we have some attributes to consider omitting which in this data I think are date, month, and year. I think it was a label more than information to calculate, at first I tried to make them into dates in date format. But after arranging them in date format and sorting them accordingly I find the no need to sort them as they have a lot of jumps in the data. So in my opinion we can consider omitting that from the variables we use.



Descriptions
Of the predictors

SD, VAR AND RANGE

```
> functions <- c(sd,var,range)
> for(var in functions) {
+   print(var)
+   print(sapply(WAUS, var, na.rm=TRUE))
+ }
function (x, na.rm = FALSE)
sqrt(var(if (is.vector(x) || is.factor(x)) x else as.double(x),
    na.rm = na.rm))
<bytecode: 0x000001d0208ef458>
<environment: namespace:stats>
          Day          Month           Year       Location        MinTemp 
    MaxTemp       Rainfall    Evaporation       Sunshine 
      8.717758       3.401519       3.359880      12.239158       5.935580 
   7.218506       5.787222       6.007833       3.840698 
   WindGustDir  WindGustSpeed     WindDir9am     WindDir3pm   WindSpeed9am   W
indSpeed3pm    Humidity9am    Humidity3pm    Pressure9am 
            NA      13.453036             NA             NA       9.407484 
   8.789955      21.188400      22.169712       7.206329 
   Pressure3pm       Cloud9am       Cloud3pm        Temp9am        Temp3pm War
merTomorrow           Date 
      7.050786       2.954104       2.796369       6.146112       7.217975 
   0.497409    1231.834858 
function (..., na.rm = FALSE)  .Primitive("range")
     Day  Month Year   Location MinTemp MaxTemp Rainfall Evaporation Sunshine
 WindGustDir WindGustSpeed WindDir9am WindDir3pm
[1,] "1"  "1"   "2007" "2"      "-6.8"   "4.9"   "0"      "0"         "0"      
 "E"         "11"          "E"         "E"        
[2,] "31" "12"  "2019" "44"     "28.9"   "45.2"  "82.2"   "133.9"     "13.6"   
 "WSW"       "96"          "WSW"       "WSW"      
     WindSpeed9am WindSpeed3pm Humidity9am Humidity3pm Pressure9am Pressure3pm
 Cloud9am Cloud3pm Temp9am Temp3pm WarmerTomorrow Date      
[1,] "0"          "0"          "2"          "1"          "989.2"      "989.4"     
 "0"       "0"       "-0.3"    "3.2"     "0"              "13838"     
[2,] "57"         "54"         "100"        "100"        "1038.9"     "1035.9"    
 "8"       "8"       "36.8"    "43.4"    "1"              "18257"     
function (..., na.rm = FALSE)  .Primitive("range")
     Day  Month Year   Location MinTemp MaxTemp Rainfall Evaporation Sunshine
 WindGustDir WindGustSpeed WindDir9am WindDir3pm
[1,] "1"  "1"   "2007" "2"      "-6.8"   "4.9"   "0"      "0"         "0"      
 "E"         "11"          "E"         "E"        
[2,] "31" "12"  "2019" "44"     "28.9"   "45.2"  "82.2"   "133.9"     "13.6"   
 "WSW"       "96"          "WSW"       "WSW"      
     WindSpeed9am WindSpeed3pm Humidity9am Humidity3pm Pressure9am Pressure3pm
 Cloud9am Cloud3pm Temp9am Temp3pm WarmerTomorrow Date      
[1,] "0"          "0"          "2"          "1"          "989.2"      "989.4"     
 "0"       "0"       "-0.3"    "3.2"     "0"              "13838"     
[2,] "57"         "54"         "100"        "100"        "1038.9"     "1035.9"    
 "8"       "8"       "36.8"    "43.4"    "1"              "18257"     
```

Above are the rest of the variable's information we got from the WAUS by looping as use sapply to get those information.

```
#2
#drop the NA's
WAUS <- WAUS[rowSums(is.na(WAUS)) == 0,]
for(var in functions) {
  print(var)
  print(sapply(WAUS, var, na.rm=TRUE))
}

WAUS <- WAUS %>% select(-Day,-Month,-Year,-Date)
WAUS$WindDir3pm = as.factor(WAUS$WindDir3pm)
WAUS$WindDir9am = as.factor(WAUS$WindDir9am)
WAUS$WindGustDir = as.factor(WAUS$WindGustDir)
WAUS$WarmerTomorrow = as.factor(WAUS$WarmerTomorrow)
```

2. Document any pre-processing required to make the data set suitable for the model fitting that follows. (1 Mark)

To answer the The code above is the pre-processing that i did to make the data set more suitable, first I drop the NA as I explained earlier in the number 1. And then, so that the variables can be processed I turn the categorical strings as a factor so that it can be processed for the statistical or even the predictions. Therefore, to conclude I just removed the variables I consider omitting earlier then remove columns with NAs and turn the categorical string into factors

3. Divide your data into a 70% training and 30% test set by adapting the following code (written for the iris data). Use your student ID as the random seed.

As we can see below we have divided our data so 70% will go to the train and 30% to test according to the seed which is my student id number.

```
#3
set.seed(31954081) #Student ID as random seed
train.row = sample(1:nrow(WAUS), 0.7*nrow(WAUS))
WAUS.train = WAUS[train.row,]
WAUS.test = WAUS[-train.row,]

nrow(WAUS.train)
nrow(WAUS.test)
```

```
> nrow(WAUS.train)
[1] 417
> nrow(WAUS.test)
[1] 180
```

So, 417 goes to WAUS.train and 180 goes to WAUS.test. The other columns are droped because they contained NAs.
Summary of the decision tree :

```
Classification tree:
tree(formula = WarmerTomorrow ~ ., data = WAUS.train)
Variables actually used in tree construction:
 [1] "Sunshine"     "Evaporation"  "Temp3pm"      "WindGustDir"  "Humidity9am"  "WindDir9am"   "Temp9am"      "WindDir3pm"
 [9] "WinTemp"      "WindSpeed3pm" "MaxTemp"      "WindGustSpeed"
Number of terminal nodes:  30
Residual mean deviance:  0.4305 = 166.6 / 387
Misclassification error rate: 0.08633 = 36 / 417
```

The graph I will got from the rest will be plotted after this page.

4. IMPLEMENT A CLASSIFICATION MODEL USING EACH OF THE FOLLOWING TECHNIQUES. FOR THIS QUESTION YOU MAY USE EACH OF THE R FUNCTIONS AT THEIR DEFAULT SETTINGS IF SUITABLE. (5 MARKS)

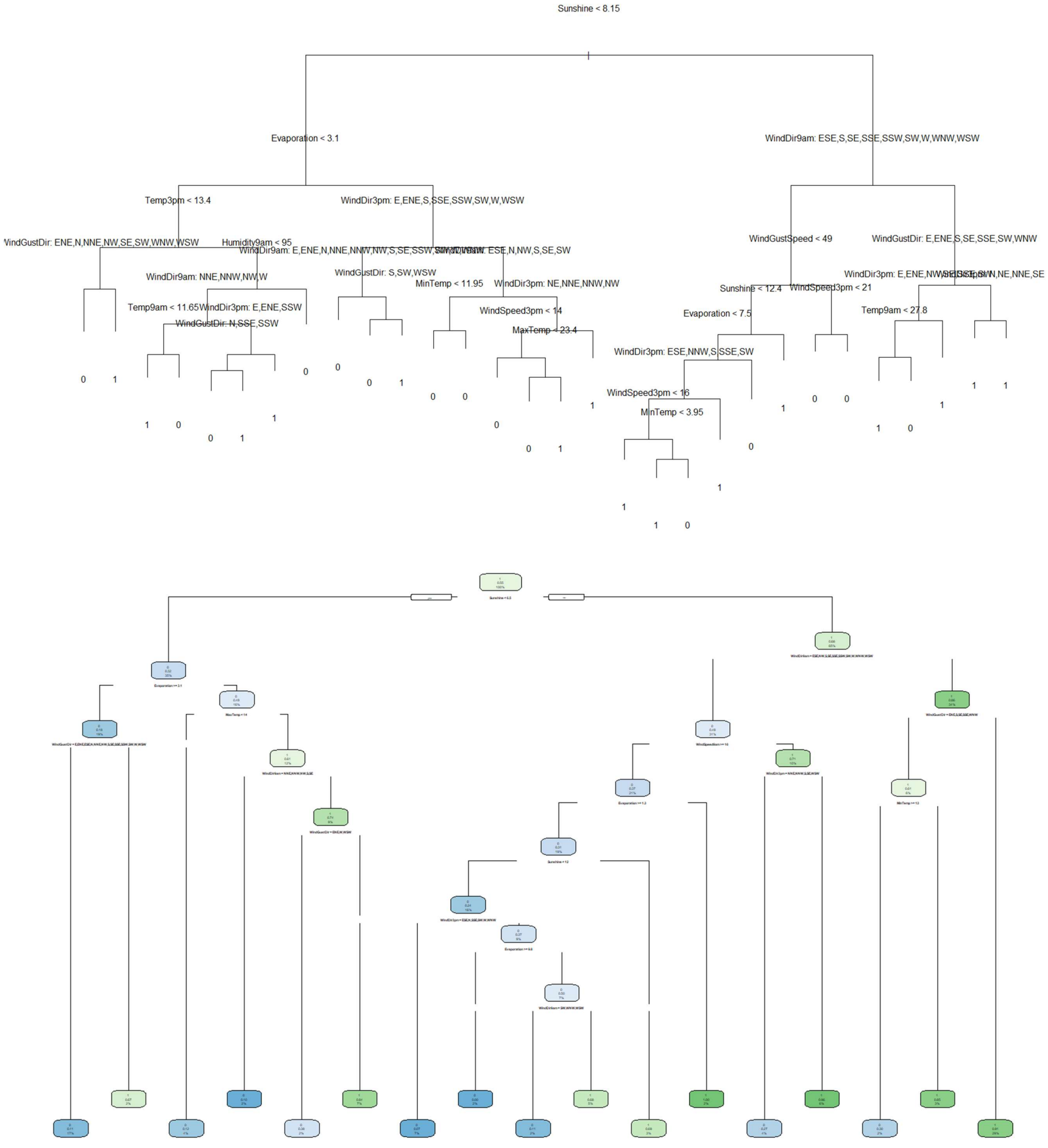
For the implementation of the model of each classification below are the codes and snippets that I use from the lecture slides.

For the implementation of the classifications model all of them I implemented by following the tutorials and do almost all the default while some settings like mfinal and etc, I do it by a simple observation then implement it to the code.

So basically, to explain it one by one decision tree is an algorithm that have conditional to classify data. While, naives bayes meaning is for those who has independent machine learning algorithm that is used in lot types of classification type. Boosting method is by training its predictors sequentially to correct its predecessor. Bagging is a high-variance machine learning algorithm. And random forest combines the output of multiple decision trees to reach a single result.

```
#4 and ROC
#https://d3cgwrxphz0fqu.cloudfront.net/38/e2/38e2bc6f81d1dd5659be73c3fb5b3a
#decision tree ##############################################
WAUS.tree =  tree(WarmerTomorrow ~., data = WAUS.train)
summary(WAUS.tree)
plot(WAUS.tree)
text(WAUS.tree, pretty = 0)
WAUS.predtree = predict(WAUS.tree, WAUS.test, type = "class")

# do predictions as probabilities and draw ROC
WAUS.pred.tree = predict(WAUS.tree, WAUS.test, type = "vector")

# computing a simple ROC curve (x-axis: fpr, y-axis: tpr)
# labels are actual values, predictors are probability of class
WAUSDpred <- prediction( WAUS.pred.tree[,2], WAUS.test$WarmerTomorrow)
WAUSDperf <- performance(WAUSDpred,"tpr","fpr")
plot(WAUSDperf)
abline(0,1)

#naive bayes ################################################
WAUS.bayes = naiveBayes(WarmerTomorrow ~. , data = WAUS.train)

# outputs as confidence levels
WAUS.predbayes = predict(WAUS.bayes, WAUS.test)
WAUSpred.bayes = predict(WAUS.bayes, WAUS.test, type = 'raw')
WAUSBpred <- prediction( WAUSpred.bayes[,2], WAUS.test$WarmerTomorrow)
WAUSBperf <- performance(WAUSBpred,"tpr","fpr")
plot(WAUSBperf, add=TRUE, col = "blueviolet")
```

```
#Bagging ####################################################
WAUS.bag <- bagging(WarmerTomorrow ~. , data = WAUS.train, mfinal=5)
WAUSpred.bag <- predict.bagging(WAUS.bag, WAUS.test)
WAUSBagpred <- prediction( WAUSpred.bag$prob[,2], WAUS.test$WarmerTomorrow)
WAUSBagperf <- performance(WAUSBagpred,"tpr","fpr")
plot(WAUSBagperf, add=TRUE, col = "blue")

#Boosting ###################################################
WAUS.Boost <- boosting(WarmerTomorrow ~. , data = WAUS.train, mfinal=10)
WAUSpred.boost <- predict.boosting(WAUS.Boost, newdata=WAUS.test)
WAUSBoostpred <- prediction( WAUSpred.boost$prob[,2], WAUS.test$WarmerTomorrow)
WAUSBoostperf <- performance(WAUSBoostpred,"tpr","fpr")
plot(WAUSBoostperf, add=TRUE, col = "red")

# Random Forest###############################################
WAUS.test <- na.omit(WAUS.test)
WAUS.train <- na.omit(WAUS.train)
WAUS.rf <- randomForest(WarmerTomorrow ~. , data = WAUS.train, na.action = na.exclude)
WAUSpred.rf <- predict(WAUS.rf, WAUS.test, type="prob")
WAUSpredrf <- predict(WAUS.rf, WAUS.test)
WAUSFpred <- prediction( WAUSpred.rf[,2], WAUS.test$WarmerTomorrow)
WAUSFperf <- performance(WAUSFpred,"tpr","fpr")
plot(WAUSFperf, add=TRUE, col = "darkgreen")
```



Both above are the plot I got from the decision tree made by plot and add text or by using rpart. We can see that we can do some classification of 1 or 0 for those.

```
#5
#Decision Tree ##########################
# do predictions as classes on tree and draw a table
t1=table(Predicted_Class = WAUS.predtree, Actual_Class = WAUS.test$WarmerTomorrow)
cat("\n#Decision Tree Confusion\n")
print(t1)

#naive bayes ############################
t2=table(Predicted_Class = WAUS.predbayes, Actual_Class = WAUS.test$WarmerTomorrow)
cat("\n#NaiveBayes Confusion\n")
print(t2)

#Random Forest ############################
t3=table(Predicted_Class = WAUSpredrf, Actual_Class = WAUS.test$WarmerTomorrow)
cat("\n#Random Forest Confusion\n")
print(t3)

#Boosting ################################
cat("\n#Boosting Confusion\n")
print(WAUSpred.boost$confusion)

#Bagging #################################
cat("\n#Bagging Confusion\n")
print(WAUSpred.bag$confusion)


#check all accuracy
accuracyDtree = round(sum(diag(t1))/sum(t1),4)
accuracyNB = round(sum(diag(t2))/sum(t2),4)
accuracyBag = round((1 - WAUSpred.bag$error), 4)
accuracyBoost = round((1 - WAUSpred.boost$error), 4)
accuracyRF = round(sum(diag(t3))/sum(t3), 4)
```

```
#Decision Tree Confusion
> print(t1)
              Actual_Class
Predicted_Class  0  1
              0 56 25
              1 31 68
>
> #naive bayes ############################
> t2=table(Predicted_Class = WAUS.predbayes, Actual_Class = WAUS.test$WarmerTomorrow)
> cat("\n#NaiveBayes Confusion\n")

#NaiveBayes Confusion
> print(t2)
              Actual_Class
Predicted_Class  0  1
              0 47 24
              1 40 69
>
> #Random Forest ############################
> t3=table(Predicted_Class = WAUSpredrf, Actual_Class = WAUS.test$WarmerTomorrow)
> cat("\n#Random Forest Confusion\n")

#Random Forest Confusion
> print(t3)
              Actual_Class
Predicted_Class  0  1
              0 47 19
              1 40 74
>
> #Boosting ################################
> cat("\n#Boosting Confusion\n")

#Boosting Confusion
> print(WAUSpred.boost$confusion)
                Observed Class
Predicted Class  0  1
              0 51 26
              1 36 67
>
> #Bagging #################################
> cat("\n#Bagging Confusion\n")

#Bagging Confusion
> print(WAUSpred.bag$confusion)
                Observed Class
Predicted Class  0  1
              0 52 26
              1 35 67
```

5. Using the test data, classify each of the test cases as ‘warmer tomorrow’ or ‘not warmer tomorrow’. Create a confusion matrix and report the accuracy of each model. (1 Mark)

EXPLANATION

Above are the confusion matrix of each classifier, to simply conclude here are the accuracy for each predictor.

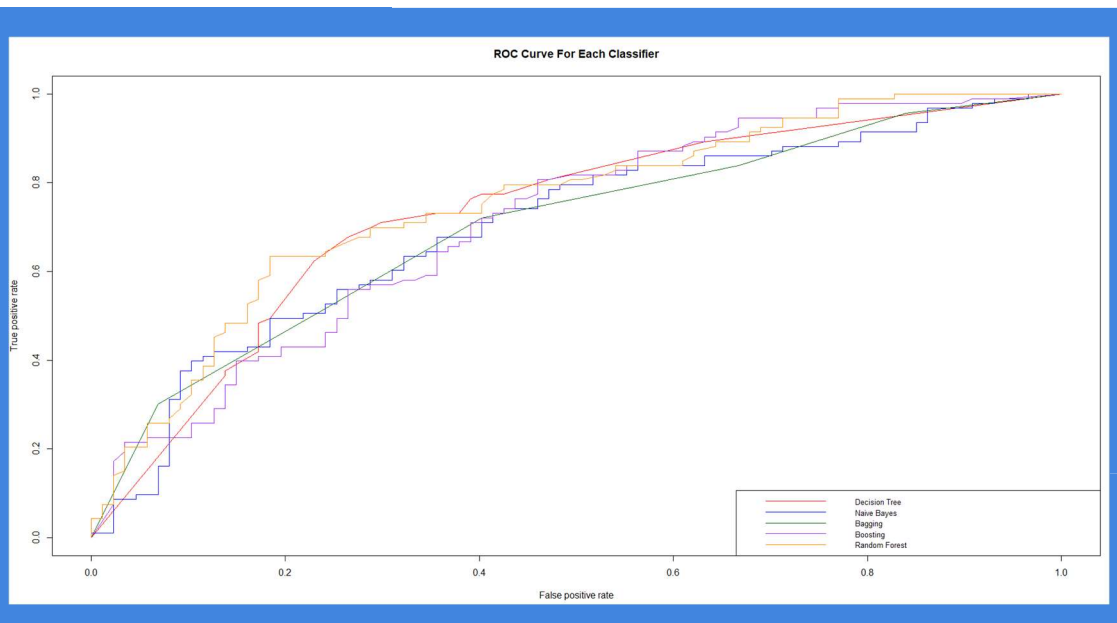
```
> accuracyDtree
[1] 0.6889
> accuracyNB
[1] 0.6444
> accuracyBag
[1] 0.6611
> accuracyBoost
[1] 0.6556
> accuracyRF
[1] 0.6722
```

From the output code, we can see that so far accuracyDtree gives of the highest accuracy followed by random forest, bagging and naïve bayes. So therefore, the best predictor so far is the decision tree in which have 68.89% accuracy then followed by random forest with 67.22% accuracy, then Boosts with 65.56% accuracy and last but not least is with the naïve bayer which he has 64.44 %

For the next question, the code will be and the result is below.

```
>
> # Plot ROC curve and calculate AUC for each classifie
r
> num_models = 5 # number of models used
> model_names = c("Decision Tree", "Naive Bayes", "Bagg
ing", "Boosting", "Random Forest")
> cust_colors = c("red", "blue", "dark green","purple",
 "dark orange") # list of colors
> preds = cbind(WAUS.pred.tree[,2], WAUSpred.bayes[,2],
 WAUSpred.bag$prob[,2], WAUSpred.boost$prob[,2], WAUSpr
ed.rf[,2]) # list of prediction prob in different model
s
> auc = c()
> for (i in 1:num_models) {
+   pred = prediction(preds[,i], WAUS.test$WarmerTomorr
ow)
+   plot(performance(pred, "tpr", "fpr"),
+        add=(i!=1), col=cust_colors[i], main="ROC Curv
e For Each Classifier") # ROC
+   cauc = performance(pred, "auc") # AUC
+   cauc2 = round(as.numeric(cauc@y.values), 4) # Round
 up to 4 decimal places
+   auc = append(auc, cauc2)
+   print(paste0(cauc@y.name," (AUC) for ", model_names
[i], ": ", cauc2))
+ }
```

6. Using the test data, calculate the confidence of predicting ‘warmer tomorrow’ for each case and construct an ROC curve for each classifier. You should be able to plot all the curves on the same axis. Use a different colour for each classifier. Calculate the AUC for each classifier. (1 Mark)



As we can see this is the plot for ROC, in which we can observe all have similar curve and shape with some a bit differences.

Visualizing the ROC curve of a classifier is useful, but in many cases this information can be reduced to a single metric, the AUC. AUC represents the area under the (ROC) curve. In general, the higher the AUC score, the better the classifier's performance for a particular task. From the result we can see that now the ROC

Here are the AUC and ROC we got from computing the code above.

```
[1] "Area under the ROC curve (AUC) for Decision Tree: 0.7311"
[1] "Area under the ROC curve (AUC) for Naive Bayes: 0.6977"
[1] "Area under the ROC curve (AUC) for Bagging: 0.6987"
[1] "Area under the ROC curve (AUC) for Boosting: 0.7085"
[1] "Area under the ROC curve (AUC) for Random Forest: 0.7518"
```

```
> compare
              accuracy    AUC
Decision tree   0.6889 0.7311
Naive Bayes     0.6444 0.6977
Bagging         0.6611 0.6987
Boostiong       0.6556 0.7085
Random forest   0.6722 0.7518
```

This is the comparation with AUC and The normal predictions

```
#Decision Tree Attribute Importance
> print(summary(WAUS.tree))

Classification tree:
tree(formula = WarmerTomorrow ~ ., data = WAUS.train)
Variables actually used in tree construction:
 [1] "Sunshine"      "Evaporation"   "Temp3pm"       "WindGustDir"   "Humidity9am"   "WindDir9am"
 [7] "Temp9am"       "WindDir3pm"    "MinTemp"       "WindSpeed3pm"  "MaxTemp"       "WindGustSpeed"
Number of terminal nodes:  30
Residual mean deviance:  0.4305 = 166.6 / 387
Misclassification error rate: 0.08633 = 36 / 417
> cat("\n#Baging Attribute Importance\n")

#Baging Attribute Importance
> print(sort(WAUS.bag$importance,decreasing = T))
  WindGustDir    WindDir9am      Sunshine    WindDir3pm   Evaporation   Humidity3pm      Cloud9am WindGustSpeed
   18.5739938    17.7928217    17.5167831    10.1498873     8.4895751     5.1483105     4.3863059     3.8023982
      MaxTemp   Pressure9am   Pressure3pm      Cloud3pm       Temp3pm       MinTemp       Temp9am   Humidity9am
    2.9998265     2.9756740     2.5403605     1.3269684     1.0860499     0.8382443     0.7771317     0.6534045
 WindSpeed9am      Location      Rainfall  WindSpeed3pm
    0.5457098     0.3965548     0.0000000     0.0000000
> cat("\n#Boosting Attribute Importance\n")

#Boosting Attribute Importance
> print(sort(WAUS.Boost$importance,decreasing = T))
   WindDir9am   WindGustDir    WindDir3pm      Sunshine   Evaporation       Temp3pm   Humidity9am       MinTemp
   17.2294257    15.6976324    13.2617633     8.9879717     6.8128064     4.6860780     4.6491733     4.6255808
     Cloud9am       MaxTemp   Humidity3pm   Pressure3pm      Temp9am  WindSpeed9am   Pressure9am WindGustSpeed
    4.1823327     4.1524560     3.8331716     3.2062273     2.3759731     2.0675132     1.9928650     0.8859768
     Location  WindSpeed3pm      Cloud3pm      Rainfall
    0.7929470     0.5601057     0.0000000     0.0000000
> cat("\n#Random Forest Attribute Importance\n")

#Random Forest Attribute Importance
> print(WAUS.rf$importance[order(-WAUS.rf$importance),])
   WindDir9am   WindGustDir    WindDir3pm      Sunshine       Temp3pm   Humidity3pm   Evaporation       MinTemp
    24.612097     20.721882     18.948179     18.868650     11.526019     11.075289     10.444076      9.861109
  Pressure9am       MaxTemp      Cloud9am   Pressure3pm   Humidity9am WindGustSpeed       Temp9am  WindSpeed3pm
     9.746269      9.608221      8.227918      8.073107      7.423717      7.160877      6.818610      6.283998
 WindSpeed9am      Cloud3pm      Location      Rainfall
     5.762795      4.547681      3.354280      2.600833
> #9
```

This are all the variables used and its level of importances

7. Create a table comparing the results in parts 5 and 6 for all classifiers. Is there a single “best” classifier? (1 Mark)

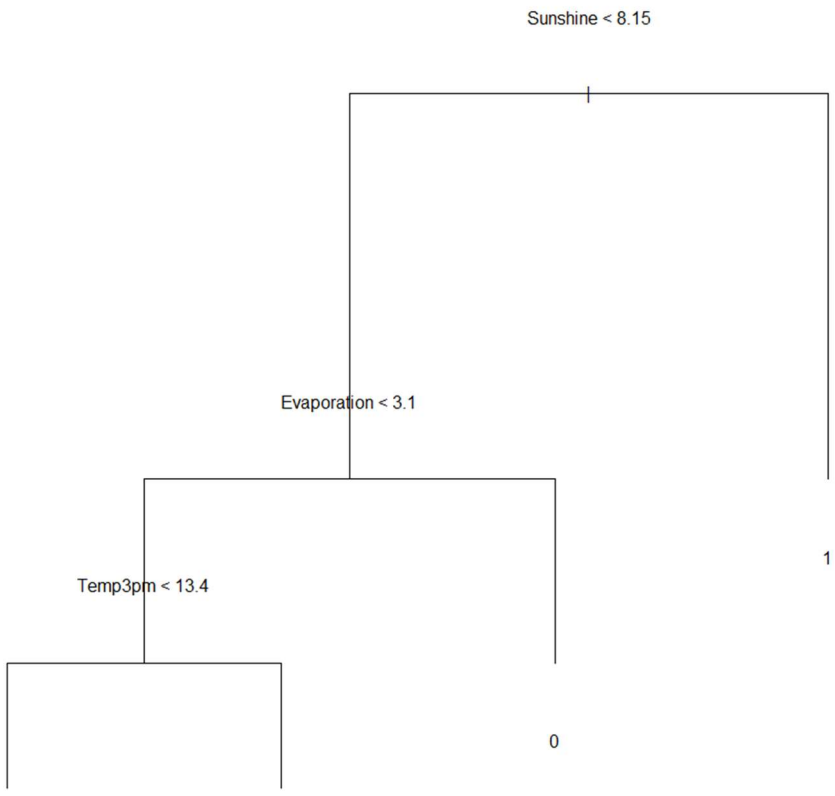
From what we can see with the top left image it shows that without AUC Decision tree should be the best but random forest which has an initial accuracy of 0.6722 and AUC 0.7518. And this is followed by the decision tree which has the initial highest which 0.6889 and have AUC for 0.7311. So, it is safe to say the random forest is the best classifiers.

8. EXAMINING EACH OF THE MODELS, DETERMINE THE MOST IMPORTANT VARIABLES IN PREDICTING WHETHER IT WILL BE WARMER TOMORROW OR NOT. WHICH VARIABLES COULD BE OMITTED FROM THE DATA WITH VERY LITTLE EFFECT ON PERFORMANCE? GIVE REASONS. (2 MARKS)

(Importace list shown above)

After examining of each of the important variables I notice that if we use Decision Tree we can notice that they chose their own variable so no need to omit and for Bagging it were Rainfall and Cloud3pm and Boost is Cloud3pm and Rainfall. Because those don’t even contribute by not having above 0.0 result. So we should consider omitting Rainfall and Windspeed3pm if we choose bagging an omit Cloud3Pm and Rainfall if we use boost.

9. **Starting with one of the classifiers you created in Part 4, create a classifier that is simple enough for a person to be able to classify whether it will be warmer tomorrow or not by hand. Describe your model, either with a diagram or written explanation. How well does your model perform, and how does it compare to those in Part 4? What factors were important in your decision? State why you chose the attributes you used. (2 Marks)**




```
> # Calculate accuracy
> pruned_dt.predict = predict(pruned_dt.fit, WAUS.test, type="class")
> pruned_dt.cm = table(actual=WAUS.test$WarmerTomorrow, predicted=pruned_dt.predic
t) # confusion matrix
> pruned_dt.cm
      predicted
actual  0  1
     0 36 51
     1 16 77
> pruned_dt.acc = round(sum(diag(pruned_dt.cm))/sum(pruned_dt.cm), 4)
> print(paste0("Accuracy for Decision Tree after pruning: ", pruned_dt.acc))
[1] "Accuracy for Decision Tree after pruning: 0.6278"
> set.seed(31954081)
```

Because we need to create a model that is enough for a person to classify easily, I chose to use the decision tree which we can see below I use the top 3 variable and it does quite well.

The most important factor of this decision is the simple and easy and its result is not bad with its 62.78% accuracy after pruning in which I use for the next number also. Also following the previous number, I chose to exclude the dates but reuse all as decision tree pruned and chose the variable.

As explained earlier decision tree is a simple algorithm that classifies based on the variable to get its classification for example in the graph beside we can see that if sunshine bigger than 8.15 it is most probably that tomorrow will be warmer then if less and its evaporation bigger than 3.1 its probably be colder then if its evaporation less than 3.1 if its temperature3pm bigger than 13.4 it will be warmer and otherwise.

The variables that are important in this case are the sunshine, evaporation and temperature3pm because after pruning the tree with its best 3, those variables are the best 3 variables we get from the decision tree algorithm after pruning.

we can see on the graph besides, we can easily classify by hand either tomorrow will be warmer or not.

```
#9
#By hand : decision tree because it categorized all the needed and can be deduct
# Visualize the decision tree (see how we can prune)
plot(print(WAUS.tree))
text(WAUS.tree, pretty=0)

fit <- rpart(WarmerTomorrow ~., data = WAUS.train, method = 'class')
rpart.plot(fit, extra = 106)

# Use K-fold cv function to generate attributes for observation use
test_dt.fit = cv.tree(WAUS.tree, FUN = prune.tree)
test_dt.fit

# Choose the smallest $dev and $k for pruning process
pruned_dt.fit = prune.misclass(WAUS.tree, best = 3)
summary(pruned_dt.fit)

# < Observation >
# You would get k = 0 and it simply means model performance won't get changed
# although the tree is simplified into one with 2 leaf nodes in the end.

# View the pruned decision tree
plot(pruned_dt.fit)
text(pruned_dt.fit, pretty=0)

# Calculate accuracy
pruned_dt.predict = predict(pruned_dt.fit, WAUS.test, type="class")
pruned_dt.cm = table(actual=WAUS.test$WarmerTomorrow, predicted=pruned_dt.predict
pruned_dt.cm
pruned_dt.acc = round(sum(diag(pruned_dt.cm))/sum(pruned_dt.cm), 4)
print(paste0("Accuracy for Decision Tree after pruning: ", pruned_dt.acc))

set.seed(31954081)
new_rf.fit = randomForest(WarmerTomorrow ~ ., data=WAUS.train, mtry = 7)
#use cv to generate attributes
#https://stackoverflow.com/questions/31637259/random-forest-crossvalidation-in-r
#rf.crossvalidation(WAUS.rf, WAUS.test, n=99, plot=TRUE, seed = 31954081 )
# Run cross validation for random forest
rain_rfcv = rfcv(trainx=WAUS.train[,c(1:20)], trainy=WAUS.train[,c(21)], step=0.
rain_rfcv$error.cv

print(new_rf.fit$importance[order(-WAUS.rf$importance),])
```

10.CREATE THE BEST TREE-BASED CLASSIFIER YOU CAN. YOU MAY DO THIS BY ADJUSTING THE PARAMETERS, AND/OR CROSS-VALIDATION OF THE BASIC MODELS IN PART 4 OR USING AN ALTERNATIVE TREE-BASED LEARNING ALGORITHM. SHOW THAT YOUR MODEL IS BETTER THAN THE OTHERS USING APPROPRIATE MEASURES. DESCRIBE HOW YOU CREATED YOUR IMPROVED MODEL, AND WHY YOU CHOSE THAT MODEL. WHAT FACTORS WERE IMPORTANT IN YOUR DECISION? STATE WHY YOU CHOSE THE ATTRIBUTES YOU USED. (3 MARKS)

With some observation from the AUC and ROC we can see that the decision tree and random forest was by far the best models that we should consider using.

```
> # Use K-fold cv function to generate attributes for observation use
> test_dt.fit = cv.tree(WAUS.tree, FUN = prune.tree)
> test_dt.fit
$size
 [1] 30 29 28 27 25 24 23 22 20 19 18 17 16 15 14 13 10  8  7  6  5  4  3  2  1

$dev
 [1] 1037.1416 1018.3488 1019.4822 1011.0487  984.6424  984.6424  975.0585  929.6171  850.5686  831.2357
[11]  808.9077  812.8614  730.6299  681.2421  617.6434  633.8499  634.6167  631.6929  565.7766  569.1289
[21]  563.0045  563.0045  513.0237  551.7072  583.4567

$k
 [1]      -Inf  6.230469  6.478556  6.657028  7.046810  7.048457  7.211603  7.783493  8.463015  8.848537
[11]  8.957748  9.156060 10.740855 12.111306 13.314010 13.612652 13.777866 14.189671 16.633360 16.661558
[21] 19.379914 19.668925 24.485431 39.618224 51.277653

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
>
> # Choose the smallest $dev and $k for pruning process
> pruned_dt.fit = prune.misclass(WAUS.tree, best = 3)
> summary(pruned_dt.fit)

Classification tree:
snip.tree(tree = WAUS.tree, nodes = c(8L, 5L, 9L, 3L))
Variables actually used in tree construction:
[1] "Sunshine"    "Evaporation" "Temp3pm"
Number of terminal nodes:  4
Residual mean deviance:  1.164 = 480.8 / 413
Misclassification error rate: 0.2662 = 111 / 417
>
```

And so first I do the cross validation for the decision tree and those are the output..

```
>
> # Calculate accuracy
> pruned_dt.predict = predict(pruned_dt.fit, WAUS.test, type="class")
> pruned_dt.cm = table(actual=WAUS.test$WarmerTomorrow, predicted=pruned_dt.predict) # confusion matrix
> pruned_dt.cm
      predicted
actual  0  1
     0 36 51
     1 16 77
> pruned_dt.acc = round(sum(diag(pruned_dt.cm))/sum(pruned_dt.cm), 4)
> print(paste0("Accuracy for Decision Tree after pruning: ", pruned_dt.acc))
[1] "Accuracy for Decision Tree after pruning: 0.6278"
>
```

Other than that as random forest was the highest in AUC, I also do a cross validation for the random forest

```
> set.seed(31954081)
> new_rf.fit = randomForest(WarmerTomorrow ~ ., data=WAUS.train, mtry = 7)
> #use cv to generate attributes
> #https://stackoverflow.com/questions/31637259/random-forest-crossvalidation-in-r
> #rf.crossValidation(WAUS.rf, WAUS.test, n=99, plot=TRUE, seed = 31954081 )
> # Run cross validation for random forest
> rain_rfcv = rfcv(trainx=WAUS.train[,c(1:20)], trainy=WAUS.train[,c(21)], step=0.8)
Warning message:
In doTryCatch(return(expr), name, parentenv, handler) :
  display list redraw incomplete
> rain_rfcv$error.cv
       20        16        13        10         8         7         5         4         3         2         1
0.2733813 0.2829736 0.2901679 0.2661871 0.3045564 0.3045564 0.3525180 0.3693046 0.3717026 0.3549161 0.3693046
>
> print(new_rf.fit$importance[order(-WAUS.rf$importance),])
   WindDir9am   WindGustDir    WindDir3pm      Sunshine       Temp3pm   Humidity3pm   Evaporation
    26.197454     22.049883     21.115491     20.836615     11.271171      9.891155     11.078543
      MinTemp   Pressure9am       MaxTemp      Cloud9am   Pressure3pm   Humidity9am WindGustSpeed
     9.465501      9.529058      7.636667      9.063516      7.223776      6.609156      6.462524
      Temp9am  WindSpeed3pm  WindSpeed9am      Cloud3pm      Location      Rainfall
     5.838610      6.458688      5.937869      3.972004      2.486434      2.304280
>
> # Calculate accuracy
> new_rf.predict = predict(new_rf.fit, WAUS.test)
> new_rf.cm = table(actual=WAUS.test$WarmerTomorrow, predicted=new_rf.predict) # confusion matrix
> new_rf.cm
      predicted
actual  0  1
     0 50 37
     1 19 74
> new_rf.acc = round(sum(diag(new_rf.cm))/sum(new_rf.cm), 4)
> print(paste0("Accuracy for Random Forest after CV: ", new_rf.acc))
[1] "Accuracy for Random Forest after CV: 0.6889"
```

And with those cross validation we got that output. Below is the comparison of both tree after and before cross validation

```
> new_summary_table
                  Decision Tree Random Forest
Initial_Accuracy         0.6889        0.6722
Improved_Accuracy        0.6278        0.6889
```

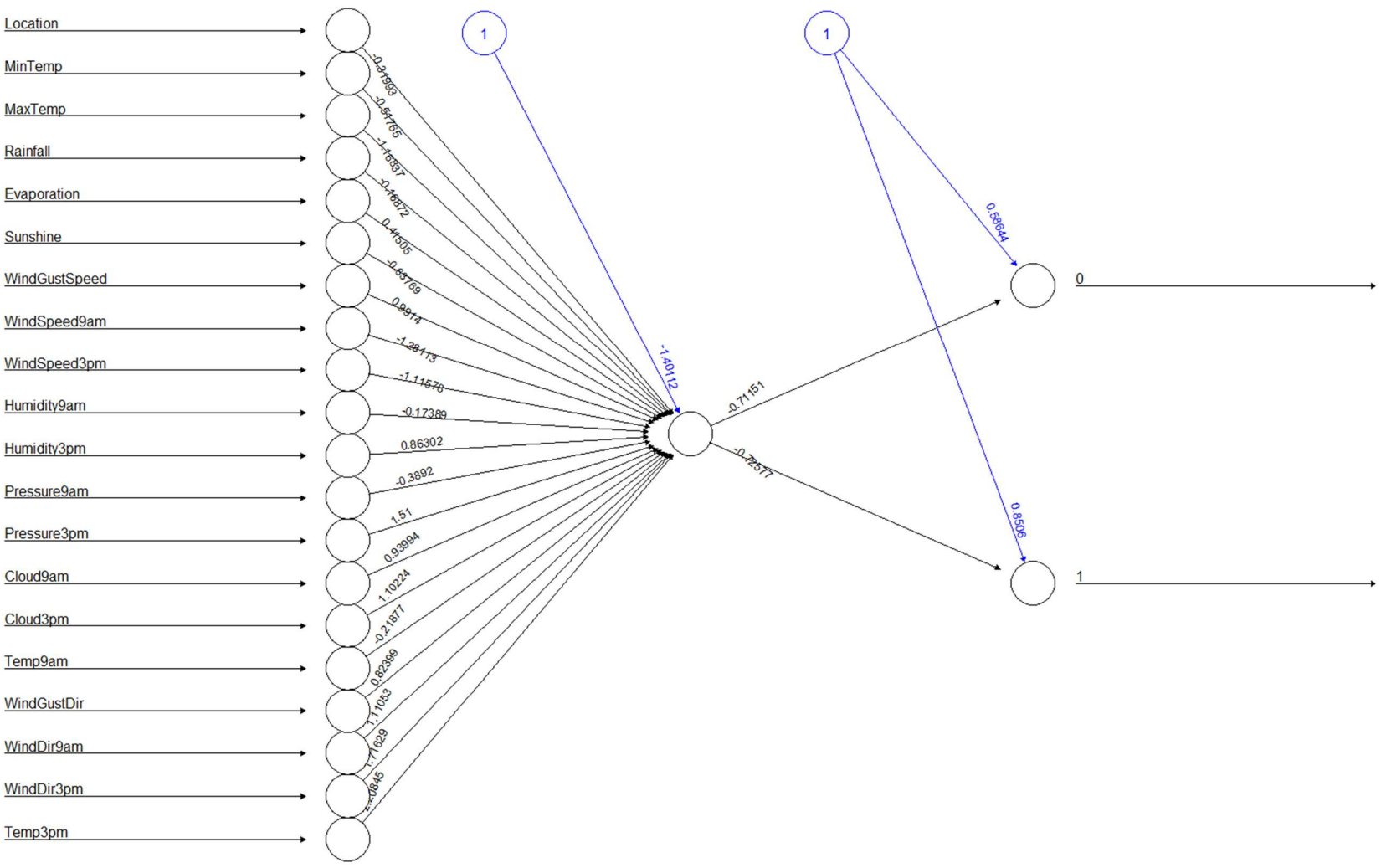
As we can see the result is that Decision tree somehow loss accuracy while random forest accuracy has improved quite significantly. This was probably the fact that because we pruned the decision tree but anyway we have improved the random forest which is the best models for our data so far according to my observation.

I didn't omit any variable except for the date related because as we observe in the important result those actually have some values even if it's a bit lower than usual.

So, in conclusion I chose the random forest method and observe it as the best model I can use.

---

11. USING THE INSIGHTS FROM YOUR ANALYSIS SO FAR, IMPLEMENT AN ARTIFICIAL NEURAL NETWORK CLASSIFIER AND REPORT ITS PERFORMANCE. COMMENT ON ATTRIBUTES USED AND YOUR DATA PREPROCESSING REQUIRED. HOW DOES THIS CLASSIFIER COMPARE WITH THE OTHERS? CAN YOU GIVE ANY REASONS? (2 MARKS)



```
> # Create confusion matrix and calculate the accuracy
> nn.cm = table(observed=new_WAUS.test$WarmerTomorrow, predicted=nn.predrdfs$WarmerTomorrow)
> nn.cm
        predicted
observed   A   B
       0   0  78
       1   0 103
> nn.acc = round(sum(diag(nn.cm))/sum(nn.cm)*100, 2)
> print(paste0("Accuracy for ANN:", nn.acc, "%"))
[1] "Accuracy for ANN:56.91%"
```

NNA benefits from the complex structure of ANN and its operators to generate new solution candidates. As a statistically superior algorithm without the effort to finetune the initial parameters, NNAs may differ from other reported optimizers. We can conclude that ANN and its special structure can be successfully used and modeled as a metaheuristic optimization technique for addressing optimization problems.

In ANN terminology, NNA is an adaptive, unsupervised method for solving optimization problems. Unsupervised at NNA means that the solution is updated by learning from the environment without any hints or information about global optimization. NNA is a single-layer self-feedback perceptron optimization technique.

Above are the neural network that I made using the code, it shows quite a low accuracy which I got through multiple trials. I tried to separate the warmer tomorrow and not warmer tomorrow but it still won't do it. And also try to normalize its non-factor variable but it still won't give seom satisfying result. For the confusion matrix I tried to observe by looking at the 1s data and not the 0s data, in case it would cause a confusion.

As usual I use the whole attributes except the date related to do the predictions, I tried to omit some of the seemingly less important one such as rainfall and etc. but it cause the accuracy to be even lower therefore, I use all variables instead to do this part.

If I were to compare this classifier to the others, I think accuracy wise to the others event though this is not perfect in my observation it would still won't be higher than the 5 classifiers we try. And won't have accuracy as high as decision tree or even random forest. Although we might have to consider if I did some error on my part on executing the codes.

So, I would still use the random forest as the best classifiers for this data.

APPENDIX

Below are the r codes I used to get the answer and reports

```
#setting up the directory
setwd("D:/MONASH/FIT3152/A2")

rm(list = ls())
WAUS <- read.csv("WarmerTomorrow2022.csv")
L <- as.data.frame(c(1:49))
set.seed(31954081) # Your Student ID is the random seed
L <- L[sample(nrow(L), 10, replace = FALSE),] # sample 10 locations
WAUS <- WAUS[(WAUS$Location %in% L),]
WAUS <- WAUS[sample(nrow(WAUS), 2000, replace = FALSE),] # sample 2000 rows

#1
#need to set date to the x axis eh sort by dates, month and year
WAUS$Date<-as.Date(with(WAUS,paste(Year,Month,Day,sep="-")),"%Y-%m-%d")
WAUS[order(as.Date(WAUS$Date, format="%Y-%m-%d")),]

library(dplyr)
library(rpart)
library(tree) # for Decision Tree
library(randomForest) # for Random Forest
library(e1071) # for Naive Bayes
library(adabag) # for Bagging and Boosting
library(neuralnet) # for ANN
library(ROCR) # for AUC and ROC
library(rpart)
library(rpart.plot)
library(corrplot)
library("factoextra")
library("FactoMineR")


#take the 0 so we know today is warmer than tmr
#1 is today(previous day) is colder than tmr(today)
#count total of one divided by all rows total then times 100
today_warmer <- WAUS %>%S
                filter(WarmerTomorrow==1)

today_warmer_percentage <- (count(today_warmer) / count(WAUS)*100)
today_warmer_percentage

warmerOrNot <- c(as.numeric(today_warmer_percentage), 100 - as.numeric(today_warmer_percentage))
labels <- c("Warmer", "Not Warmer")
pie(warmerOrNot,labels)
```

```
#summary

#https://www.statmethods.net/stats/descriptives.html#:~:text=R%20provides%20a%20wide%20range%20of%20functions%20for,provide%20a%20range%20of%20descriptive%20statistics%20at%20once.

summary(WAUS)



#install.packages('pastecs')

#require(pastecs)

#stat.desc(WAUS)



#omit location and date etc. from the predictions as it is probably not affecting but we still need for labels and info purpose



functions <- c(sd,var,range)

for(var in functions) {

  print(var)

  print(sapply(WAUS, var, na.rm=TRUE))

}



#notice there are many NA's in the data

#the NA's causing problem with the data so there are some calculations error



#2

#drop the NA's

WAUS <- WAUS[rowSums(is.na(WAUS)) == 0,]

for(var in functions) {

  print(var)

  print(sapply(WAUS, var, na.rm=TRUE))

}



WAUS <- WAUS %>% select(-Day,-Month,-Year,-Date)

WAUS$WindDir3pm = as.factor(WAUS$WindDir3pm)

WAUS$WindDir9am = as.factor(WAUS$WindDir9am)

WAUS$WindGustDir = as.factor(WAUS$WindGustDir)

WAUS$WarmerTomorrow = as.factor(WAUS$WarmerTomorrow)



#3

set.seed(31954081) #Student ID as random seed

train.row = sample(1:nrow(WAUS), 0.7*nrow(WAUS))

WAUS.train = WAUS[train.row,]

WAUS.test = WAUS[-train.row,]



nrow(WAUS.train)

nrow(WAUS.test)





#4 and ROC

#https://d3cgwrxphz0fqu.cloudfront.net/38/e2/38e2bc6f81d1dd5659be73c3fb5b3a87888fb811?response-content-disposition=inline%3Bfilename%3D%22FIT3152%20Tutorial%2009%20%2B%20Solutions.pdf%22&response-content-type=application%2Fpdf&Expires=1652663160&Signature=VCnF0iMA2B2hAeiNuf8NMNKcvtMqICBa~~tWygIOT3l30vlaIOTDZ0Zttp1aoXqpNdpsS6431Hk14K2RU0E-JpldKpsVEKzaEDyEsWVCeKnQg7sI3JKB0Nshrztqavrwp QZe1k33Ol~egKwM-nUQ6Js55LBrbz4h6AIm1y7awIGLmZGEtlD0h3t0YsMQ5RRRnez-
```

xCwc6EL6bov4ZdhT4u8H19Y3zSXDDFAzQ1qEMlfAi1nr4U9t50UFoQx0bAW66rFIlyI-XLcMtzu1vOFSSrIshDByH56TRSia09OsEqUEb-ujeluTIlsc58UD9Ra893nWUzfR2TIWaOxSQOwyUg__&Key-Pair-Id=APKAJRIEZFHR4FGFTJHA

```
#decision tree ################################################

WAUS.tree =  tree(WarmerTomorrow ~., data = WAUS.train)

summary(WAUS.tree)

plot(WAUS.tree)

text(WAUS.tree, pretty = 0)

WAUS.predtree = predict(WAUS.tree, WAUS.test, type = "class")


# do predictions as probabilities and draw ROC

WAUS.pred.tree = predict(WAUS.tree, WAUS.test, type = "vector")


# computing a simple ROC curve (x-axis: fpr, y-axis: tpr)

# labels are actual values, predictors are probability of class

WAUSDpred <- prediction( WAUS.pred.tree[,2], WAUS.test$WarmerTomorrow)
WAUSDperf <- performance(WAUSDpred,"tpr","fpr")

plot(WAUSDperf)

abline(0,1)


#naive bayes ###################################################

WAUS.bayes = naiveBayes(WarmerTomorrow ~. , data = WAUS.train)


# outputs as confidence levels

WAUS.predbayes = predict(WAUS.bayes, WAUS.test)

WAUSpred.bayes = predict(WAUS.bayes, WAUS.test, type = 'raw')

WAUSBpred <- prediction( WAUSpred.bayes[,2], WAUS.test$WarmerTomorrow)

WAUSBperf <- performance(WAUSBpred,"tpr","fpr")

plot(WAUSBperf, add=TRUE, col = "blueviolet")


summary(WAUS.predbayes)


#Bagging ######################################################

WAUS.bag <- bagging(WarmerTomorrow ~. , data = WAUS.train, mfinal=5)

WAUSpred.bag <- predict.bagging(WAUS.bag, WAUS.test)


WAUSBagpred <- prediction( WAUSpred.bag$prob[,2], WAUS.test$WarmerTomorrow)

WAUSBagperf <- performance(WAUSBagpred,"tpr","fpr")

plot(WAUSBagperf, add=TRUE, col = "blue")


#Boosting ######################################################

WAUS.Boost <- boosting(WarmerTomorrow ~. , data = WAUS.train, mfinal=10)

WAUSpred.boost <- predict.boosting(WAUS.Boost, newdata=WAUS.test)

WAUSBoostpred <- prediction( WAUSpred.boost$prob[,2], WAUS.test$WarmerTomorrow)

WAUSBoostperf <- performance(WAUSBoostpred,"tpr","fpr")

plot(WAUSBoostperf, add=TRUE, col = "red")


# Random Forest#################################################
```

```
WAUS.test <- na.omit(WAUS.test)

WAUS.train <- na.omit(WAUS.train)

WAUS.rf <- randomForest(WarmerTomorrow ~. , data = WAUS.train, na.action = na.exclude)

WAUSpred.rf <- predict(WAUS.rf, WAUS.test, type="prob")

WAUSpredrf <- predict(WAUS.rf, WAUS.test)

WAUSFpred <- prediction( WAUSpred.rf[,2], WAUS.test$WarmerTomorrow)

WAUSFperf <- performance(WAUSFpred,"tpr","fpr")

plot(WAUSFperf, add=TRUE, col = "darkgreen")



#5

#Decision Tree ##########################

# do predictions as classes on tree and draw a table

t1=table(Predicted_Class = WAUS.predtree, Actual_Class = WAUS.test$WarmerTomorrow)

cat("\n#Decision Tree Confusion\n")

print(t1)


#naive bayes ############################

t2=table(Predicted_Class = WAUS.predbayes, Actual_Class = WAUS.test$WarmerTomorrow)

cat("\n#NaiveBayes Confusion\n")

print(t2)


#Random Forest ############################

t3=table(Predicted_Class = WAUSpredrf, Actual_Class = WAUS.test$WarmerTomorrow)

cat("\n#Random Forest Confusion\n")

print(t3)


#Boosting #################################

cat("\n#Boosting Confusion\n")

print(WAUSpred.boost$confusion)


#Bagging ##################################

cat("\n#Bagging Confusion\n")

print(WAUSpred.bag$confusion)



#check all accuracy

accuracyDtree = round(sum(diag(t1))/sum(t1),4)

accuracyNB = round(sum(diag(t2))/sum(t2),4)

accuracyBag = round((1 - WAUSpred.bag$error), 4)

accuracyBoost = round((1 - WAUSpred.boost$error), 4)

accuracyRF = round(sum(diag(t3))/sum(t3), 4)


accuracyDtree

accuracyNB

accuracyBag

accuracyBoost
```

#6

```
# Plot ROC curve and calculate AUC for each classifier

num_models = 5 # number of models used

model_names = c("Decision Tree", "Naive Bayes", "Bagging", "Boosting", "Random Forest")

cust_colors = c("red", "blue", "dark green","purple", "dark orange") # list of colors

preds = cbind(WAUS.pred.tree[,2], WAUSpred.bayes[,2], WAUSpred.bag$prob[,2], WAUSpred.boost$prob[,2], WAUSpred.rf[,2]) # list of
prediction prob in different models

auc = c()

for (i in 1:num_models) {

  pred = prediction(preds[,i], WAUS.test$WarmerTomorrow)

  plot(performance(pred, "tpr", "fpr"),
       add=(i!=1), col=cust_colors[i], main="ROC Curve For Each Classifier") # ROC

  cauc = performance(pred, "auc") # AUC

  cauc2 = round(as.numeric(cauc@y.values), 4) # Round up to 4 decimal places

  auc = append(auc, cauc2)

  print(paste0(cauc@y.name," (AUC) for ", model_names[i], ": ", cauc2))

}

legend("bottomright", legend=c("Decision Tree", "Naive Bayes", "Bagging", "Boosting", "Random Forest"),col=c("red", "blue", "dark
green","purple", "dark orange"), lty=1, cex=0.8)
```

```
#if add manually one by one

DTauc = performance(WAUSDpred, "auc")

AUC_DT = round(as.numeric(DTauc@y.values),4)

AUC_DT


NBauc = performance(WAUSBpred, "auc")

AUC_NB = round(as.numeric(NBauc@y.values),4)

AUC_NB


BAGauc = performance(WAUSBagpred, "auc")

AUC_BAG = round(as.numeric(BAGauc@y.values),4)

AUC_BAG


BOOSTauc = performance(WAUSBoostpred, "auc")

AUC_BOOST = round(as.numeric(BOOSTauc@y.values),4)

AUC_BOOST


RFauc = performance(WAUSFpred, "auc")

AUC_RF = round(as.numeric(RFauc@y.values),4)

AUC_RF
```

```
legend("bottomright", legend=c("decision tree", "Naive Bayes", "Bagging", "Boosting", "Random forest"),col=c("black", "blueviolet","blue",
"red", "darkgreen"), lty=1, cex=0.8)



#7

accuracy = c(accuracyDtree, accuracyNB, accuracyBag, accuracyBoost, accuracyRF)

AUC = c(AUC_DT, AUC_NB, AUC_BAG, AUC_BOOST, AUC_RF)

compare = cbind(accuracy,AUC)

rownames(compare) <- c("Decision tree", "Naive Bayes", "Bagging", "Boostiong", "Random forest")

compare


#8

#Attribute importance ####

cat("\n#Decision Tree Attribute Importance\n")

print(summary(WAUS.tree))

cat("\n#Baging Attribute Importance\n")

print(sort(WAUS.bag$importance,decreasing = T))

cat("\n#Boosting Attribute Importance\n")

print(sort(WAUS.Boost$importance,decreasing = T))

cat("\n#Random Forest Attribute Importance\n")

print(WAUS.rf$importance[order(-WAUS.rf$importance),])

# Cannot determine the importance of variables used in Naive Bayes as it only calculates the probabilities of each variable given



#9

#By hand : decision tree because it categorized all the needed and can be deducted by human eyes

# Visualize the decision tree (see how we can prune)

plot(print(WAUS.tree))

text(WAUS.tree, pretty=0)


fit <- rpart(WarmerTomorrow ~., data = WAUS.train, method = 'class')

rpart.plot(fit, extra = 106)


# Use K-fold cv function to generate attributes for observation use

test_dt.fit = cv.tree(WAUS.tree, FUN = prune.tree)

test_dt.fit


# Choose the smallest $dev and $k for pruning process

pruned_dt.fit = prune.misclass(WAUS.tree, best = 3)

summary(pruned_dt.fit)


# < Observation >

# You would get k = 0 and it simply means model performance won't get changed

# although the tree is simplified into one with 2 leaf nodes in the end.


# View the pruned decision tree

plot(pruned_dt.fit)

text(pruned_dt.fit, pretty=0)
```

```
# Calculate accuracy
pruned_dt.predict = predict(pruned_dt.fit, WAUS.test, type="class")
pruned_dt.cm = table(actual=WAUS.test$WarmerTomorrow, predicted=pruned_dt.predict) # confusion matrix
pruned_dt.cm
pruned_dt.acc = round(sum(diag(pruned_dt.cm))/sum(pruned_dt.cm), 4)
print(paste0("Accuracy for Decision Tree after pruning: ", pruned_dt.acc))

set.seed(31954081)
new_rf.fit = randomForest(WarmerTomorrow ~ ., data=WAUS.train, mtry = 7)
#use cv to generate attributes
#https://stackoverflow.com/questions/31637259/random-forest-crossvalidation-in-r
#rf.crossValidation(WAUS.rf, WAUS.test, n=99, plot=TRUE, seed = 31954081 )
# Run cross validation for random forest
rain_rfcv = rfcv(trainx=WAUS.train[,c(1:20)], trainy=WAUS.train[,c(21)], step=0.8)
rain_rfcv$error.cv

print(new_rf.fit$importance[order(-WAUS.rf$importance),])

# Calculate accuracy
new_rf.predict = predict(new_rf.fit, WAUS.test)
new_rf.cm = table(actual=WAUS.test$WarmerTomorrow, predicted=new_rf.predict) # confusion matrix
new_rf.cm
new_rf.acc = round(sum(diag(new_rf.cm))/sum(new_rf.cm), 4)
print(paste0("Accuracy for Random Forest after CV: ", new_rf.acc))

# < Observation >
# When doing sampling, the number of variables don't really give significant differences.

# Generate summary table for comparison purpose
new_acc = c(pruned_dt.acc, new_rf.acc)
new_summary_table = rbind(Initial_Accuracy=c(accuracyDtree,accuracyRF), Improved_Accuracy=new_acc)
colnames(new_summary_table) = model_names[c(1,5)]
new_summary_table

#11
# Make indicator
new_WAUS = WAUS # make a copy of current dataset to avoid modification on it
new_WAUS$TmrNotWarmer = new_WAUS$WarmerTomorrow == 0
new_WAUS$TmrWarmer = new_WAUS$WarmerTomorrow == 1
#location_categorical_vars = model.matrix(~ Location + 0, data=new_WAUS) # "+0" is to solve the problem that one of the variables is being
ignored

# Exclude some unimportant variables. change this
#new_WAUS = new_WAUS[,-c(0:6,8,11:19)]

# Normalise data (0 - 1)
normalise_ = function(col){
```

```
  return((col - min(col))/(max(col) - min(col)))
}

# Apply normalise function to all predictors that are not categorical
predictors = apply(new_WAUS[,c(1:6,8,11:19)], 2, normalise_)
new_WAUS = cbind(predictors, new_WAUS[,c(7,9:10,20:23)])

# Check the attribute of each variable
summary(new_WAUS)

# Split data into train and test
set.seed(31954081) #Student ID as random seed
train.row = sample(0:nrow(new_WAUS), 0.7*nrow(new_WAUS)) # Split into 70% and 30%
new_WAUS.train = new_WAUS[train.row,] # Assign 70% to train data
new_WAUS.test = new_WAUS[-train.row,] # Assign 30% to test data

# 11 Build ANN
set.seed(31954081)
new_WAUS.train$Location <- as.numeric(new_WAUS.train$Location)
new_WAUS.train$TmrNotWarmer <- as.numeric(new_WAUS.train$TmrNotWarmer)
new_WAUS.train$TmrWarmer <- as.numeric(new_WAUS.train$TmrWarmer)
new_WAUS.train$WindDir3pm <- as.numeric(new_WAUS.train$WindDir3pm)
new_WAUS.train$WindDir9am <- as.numeric(new_WAUS.train$WindDir9am)
new_WAUS.train$WindGustDir <- as.numeric(new_WAUS.train$WindGustDir)

nn.fit = neuralnet(WarmerTomorrow~ . -TmrWarmer -TmrNotWarmer, new_WAUS.train, hidden = 1, linear.output=F)
#nn.fit = neuralnet(WarmerTomorrow~ MaxTemp + Sunshine + Pressure9am + Temp9am + Temp3pm, new_WAUS.train, hidden = 1,
linear.output=F)

# Visualize ANN
plot(nn.fit)

# Print the weights
nn.fit$result.matrix

# Make prediction using test set
new_WAUS.test$TmrNotWarmer <- as.numeric(new_WAUS.test$TmrNotWarmer)
new_WAUS.test$TmrWarmer <- as.numeric(new_WAUS.test$TmrWarmer)
new_WAUS.test$WindDir3pm <- as.numeric(new_WAUS.test$WindDir3pm)
new_WAUS.test$WindDir9am <- as.numeric(new_WAUS.test$WindDir9am)
new_WAUS.test$WindGustDir <- as.numeric(new_WAUS.test$WindGustDir)

# Print the weights
nn.fit$result.matrix

# Make prediction using test set
nn.predict = compute(nn.fit, new_WAUS.test[,-c(21:23)])
```

```
# Make prediction
nn.predr = round(nn.predict$net.result, 0)
nn.predrdf = as.data.frame(as.table(nn.predr))

# Remove rows classified 0 - leave only classified 1 - check by the
nn.predrdfs = nn.predrdf[!nn.predrdf$Freq==0,]
nn.predrdfs$Freq = NULL
colnames(nn.predrdfs) = c("Obs", "WarmerTomorrow")
nn.predrdfs = nn.predrdfs[order(nn.predrdfs$Obs),]

# Create confusion matrix and calculate the accuracy
nn.cm = table(observed=new_WAUS.test$WarmerTomorrow, predicted=nn.predrdfs$WarmerTomorrow)
nn.cm
nn.acc = round(sum(diag(nn.cm))/sum(nn.cm)*100, 2)
print(paste0("Accuracy for ANN:", nn.acc, "%"))



#IMPORTANT NN
#new_WAUS.train$WarmerTomorrow <- as.numeric(new_WAUS.train$WarmerTomorrow)
#M = cor(new_WAUS.train[,-c(22:23)])
#corrplot(M, order = 'hclust', addrect = 2)
#M[21,]

#If want to try pca
#https://bioinfo4all.wordpress.com/2021/01/31/tutorial-6-how-to-do-principal-component-analysis-pca-in-r/
```